to solicit user input in DOS 5. We've published two versions of the DEBUG script used to create this program in past issues of *Inside DOS*. The version shown in Figure B appeared in the September 1990 issue; the version shown in Figure C appeared in the December 1990 issue.

Figure B

```
e100
b4 08 cd 21 24 df 3c 59 75 04 b0 03 eb 0e 3c 4e
e110
75 04 b0 02 eb 06 3c 1b 75 06 b0 01 b4 4c cd 21
e120
b2 07 b4 02 cd 21 eb d8
n keypress.com
rcx
28
w
q
```

If you use DOS 5, you can create KEYPRESS.COM by using this DEBUG script . . .

Figure C

```
n keypress.com
e 100 ba 82 00 8b fa 8a 4d fe
e 108 8b f1 4e fe c9 7f 05 2b
e 110 c0 eb 56 90 8a 9c 81 00
e 118 80 fb 5d 74 09 4e 7f f4
e 120 e8 77 00 eb 44 90 8b fe
e 128 4f 4e 74 0e 8a 9c 81 00
e 130 80 fb 5b 75 f4 4e 7e 05
e 138 8b ce e8 5d 00 46 46 8b
e 140 ce e8 4f 00 e8 5b 00 73
e 148 05 e8 3e 00 eb f3 8a c3
e 150 8a 9c 81 00 e8 4b 00 72
e 158 04 3a c3 74 29 46 3b f7
e 160 7e ee 8b f1 e8 23 00 eb
e 168 d8 e8 27 00 e8 33 00 8a
e 170 c3 3c 19 75 02 eb 0f 3c
e 178 0e 75 02 eb 09 3c 1b 74
e 180 05 e8 06 00 eb e3 b4 4c
e 188 cd 21 52 b2 07 b4 02 cd
e 190 21 5a c3 b4 08 cd 21 8a
e 198 d8 c3 b4 40 bb 01 00 cd
e 1a0 21 c3 80 fb 1b 74 1c 80
e 1a8 fb 61 7c 0a 80 fb 7a 7f
e 1b0 14 80 eb 20 eb 0a 80 fb
e 1b8 41 7c 0a 80 fb 5a 7f 05
e 1c0 80 eb 40 f8 c3 f9 c3
rcx
c7
w
q
(blank line)
```

. . . or this DEBUG script.

After creating either script in the DOS Editor or an ASCII-compatible word processor and saving it under the name KEYPRESS.SCR, you use the command

```
C:\>debug < keypress.scr
```

to create KEYPRESS.COM.

If you used the script shown in Figure B to create KEYPRESS.COM, you need to replace the highlighted text in Figure A with

```
echo MATCHING THE SPECIFICATION %1 (y/n)?
keypress
if errorlevel 3 goto :KILL
if errorlevel 2 goto :SKIP
```

If you used the script shown in Figure C to create KEYPRESS.COM, you need to replace the highlighted text in Figure A with

```
echo MATCHING THE SPECIFICATION %1 (y/n)?
keypress [yn]
if errorlevel 25 goto :KILL
if errorlevel 14 goto :SKIP
```

How WHATSIS.BAT works

As with most batch files, WHATSIS.BAT begins with @ECHO OFF, a command that prevents DOS from displaying each command in the batch file. The next three REM statements explain the function of the batch file. Then, the CLS statement initializes the batch file with a blank screen.

Once the screen is clear, the line

```
if "%1"=="" goto :WHOOPS
```

checks if you included a file specification when you issued the command to run the batch file. If you didn't, control branches to the WHOOPS routine:

```
:WHOOPS
echo PLEASE RUN WHATSIS.BAT AGAIN, USING THE SYNTAX
echo.
echo whatsis path\filename.ext
echo.
echo.
echo THE FILENAME AND EXTENSION MAY INCLUDE WILDCARDS
echo SUCH AS
echo.
echo whatsis works\*.tmp
echo.
echo OR
echo.
echo whatsis works\wk*.tmp
echo.
echo OR
echo.
echo whatsis works\wk*.*
echo.
echo.
echo BATCH FILE TERMINATING
goto :END
```

This routine simply echoes instructions for running the batch file, tells you the batch file is terminating, and then jumps to the END routine, which ends the batch file.

If you included a file specification when you ran the batch file, the line

```
for %%a in (%1) do type %%a >> whatsis.fyl
```

looks for files matching the file specification and appends their contents to the WHATSIS.FYL file. Then, the lines

```
copy whatsis.fyl whatsis.xxx > nul
del whatsis.fyl
if not exist whatsis.xxx goto :NONE
```

check if any files match the file specification. The first line copies the WHATSIS.FYL file to the WHATSIS.XXX file and suppresses the *1 File(s) Copied* message, the second line deletes WHATSIS.FYL, and the third line checks to see if WHATSIS.XXX exists. If no files match the file specification, WHATSIS.FYL will be empty and DOS won't copy it. Therefore, WHATSIS.XXX won't exist, indicating no files matched the file specification. In this case, control branches to the NONE routine:

```
:NONE
echo NO FILES MATCHING THE SPECIFICATION %1
echo WERE FOUND
goto :END
```

This routine echoes a message that no files matched the file specification and then jumps to the END routine.

If files do match the file specification, however, the WHATSIS.XXX file will exist. The line

```
type whatsis.xxx ¦ more
```

displays the contents of WHATSIS.XXX—the appended files that matched the file specification—one screen at a time, allowing you to check whether you want to save the files or delete them.

After the last screenful appears, the lines

```
echo.
pause
```

insert a blank line after the last line of text and then prompt you to press any key to continue. Once you press a key, the lines

```
del whatsis.xxx
cls
echo WOULD YOU LIKE ME TO DELETE ALL THE FILES
choice MATCHING THE SPECIFICATION %1 /C:yn
```

delete the WHATSIS.XXX file, clear the screen, and then ask you whether you want to delete the files. (In the DOS 5 version of WHATSIS.BAT, the lines

```
echo WOULD YOU LIKE ME TO DELETE ALL THE FILES
echo MATCHING THE SPECIFICATION %1 (y/n)?
keypress ([yn])
```

ask you if you want to delete the files.)

The line

```
if errorlevel m goto :SKIP
```

(where $m = 2$ in the DOS 6 version and $m = 2$ or 14 in the DOS 5 version) branches control to the SKIP routine if you choose not to delete the files. The SKIP routine

```
:SKIP
echo.
echo.
echo NO FILES DELETED
goto :END
```

prints two blank lines and a message that no files were deleted. Then, it branches to the END routine to end the batch file.

If you press Y to delete the files, the line

```
if errorlevel n goto :KILL
```

(where $n = 1$ in the DOS 6 version and $n = 3$ or 25 in the DOS 5 version) branches control to the KILL routine:

```
:KILL
dir %1 ¦ find "(s)" > zzyyxxq.xxx
echo           DELETED >> zzyyxxq.xxx
del %1
echo.
echo.
type zzyyxxq.xxx
del zzyyxxq.xxx > nul
```

DIR %1 lists the files matching the file specification. Next, ¦ FIND "(s)" finds the line

```
        x file(s)      y bytes
```

in the directory listing and redirects this line to the ZZYYXXQ.XXX file. The line that follows appends the word *DELETED*, padded by spaces, to the file ZZYYXXQ.XXX. Then, DEL %1 deletes the specified files. The next two ECHO commands print two blank lines on the screen, and the TYPE command types the contents of ZZYYXXQ.XXX:

```
        x file(s)       y bytes
          DELETED
```

Finally, the last line in this routine deletes the file ZZYYXXQ.XXX . Once the KILL routine completes execution, the END routine ends the batch file.

Conclusion

While disk housekeeping is always a good idea, you should be careful not to delete files unless you're sure you don't need them. The WHATSIS batch file we presented in this article provides the safety feature you need to ensure that this doesn't happen.

Creating a DOS 6 emergency boot disk

By Gregory L. Harris

Years ago, when you upgraded your DOS versions, you probably made backup copies of your new DOS disks. If your PC crashed, you could reboot from your backup disks even if the faulty computer's boot sector was corrupted due to disk error or virus activity.

But beginning with DOS 5, many PCs came with DOS—and often Windows—preinstalled. Of course, the DOS disks also came in the box, but you might have misplaced them or you might not have them handy. Also, both DOS 5 and DOS 6 occupy several disks with compressed files, which makes recovering your DOS setup more difficult.

To avoid unnecessary hassle, it's a good idea to make an emergency boot disk for DOS and keep a copy of that disk near your PC. In this article, we'll suggest several DOS 6 programs and utilities that you should include on your boot disk. Having a minimal DOS configuration on disk can help you recover from disaster.

First things first

Before you copy your DOS files onto your emergency boot disk, you should format the disk by issuing the command

```
C:\>format a: /f:n /s
```

where n is the capacity of your disk. Then, you can use the SYS command to transfer the system files onto the disk:

```
C:\>sys c:\ a:
```

Make sure the disk you're using is new and error free. This suggestion may seem obvious, but you don't want to take any chances with your emergency disk failing when you need it most. We also recommend using a different colored disk—such as a red one—to distinguish the boot disk from others at a glance.

Once you've formatted your emergency disk and transferred the system files, you should copy the files we'll list in a moment. Most software licensing agreements let you make copies for backup purposes; you should check yours to ensure you're making a legal copy.

Assuming your backup disk is a 720-Kb floppy, we recommend you copy the following DOS 6 files onto it:

- ATTRIB.EXE (11,165 bytes) lets you change a file's system attributes; this file is useful if, for example, your AUTOEXEC.BAT file is flawed but its attribute is read only.
- CHKDSK.EXE (12,907 bytes) helps you determine if your hard disk has bad sectors that led to the system crash.
- FDISK.EXE (29,333 bytes) can help you recover from a disk partition error.
- FORMAT.COM (22,717 bytes) might be your only hope if you need to reformat your hard disk following a crash.
- MEM.EXE (32,150 bytes) is a useful utility that can help you track down any memory conflicts that cause problems.
- MORE.COM (2,546 bytes) is useful for examining long text files without employing a text editor. If an error in your AUTOEXEC.BAT or CONFIG.SYS file prevents your hard disk from booting, you might need to use MORE to find the offending line if the files are more than 25 lines long.
- MOVE.EXE (17,823 bytes) is useful for transferring corrupted files (or those you suspect are corrupted) to a directory that isn't in your system's PATH; doing so lets you experiment without deleting the files.
- RESTORE.EXE (38,294 bytes) alone can allow you to restore your backed-up data and get your computer up and running again.
- SETVER.EXE (12,015 bytes) lets you run any version-sensitive applications your system might have.
- SUBST.EXE (18,478 bytes) lets you substitute in your AUTOEXEC.BAT file a drive letter for a directory path. If your AUTOEXEC.BAT file contains this command, you'll want to ensure you have a backup copy of the program.
- SYS.COM (9,379 bytes) lets you make a disk bootable—even a hard disk, if necessary.
- UNDELETE.EXE (26,420 bytes) allows you to rescue any vital files that might have been accidentally deleted, as long as its data hasn't been overwritten.
- XCOPY.EXE (15,820 bytes) lets you copy large numbers of files more efficiently.
- If your system uses a DoubleSpace drive, you must ensure that the file DBLSPACE.BIN (51,214 bytes) resides on the boot disk. The DOS 6 SYS command should install this file automatically if it's needed, but Windows 3.1 and some other programs won't. Therefore, you should always double-check. If this file is absent, you won't be able to access your compressed drive.

These files, plus the DOS system files, occupy 502,761 bytes of disk space with DBLSPACE.BIN, 451,547 without. You should also copy your system's AUTOEXEC.BAT and CONFIG.SYS files onto the disk. Then, you should write-protect the disk and put it in a safe place.

Editing out any flaws

In case a flaw in your system's AUTOEXEC.BAT or CONFIG.SYS file prevents your computer from booting, you'll want to be able to edit those files after booting from a floppy. If you have a system that runs DOS 5 or earlier, copy its EDLIN.EXE line editor (13,929 bytes under DOS 5) onto your boot disk. This small, no-frills editor saves disk space and still lets you get the job done.

Alternatively, you could copy your favorite text editor onto another floppy disk or onto your bootable floppy, if there's room. If you use the DOS 6 EDIT program, don't forget that it requires both EDIT.COM (413 bytes) and QBASIC.EXE (194,309 bytes).

A bootable floppy disk can be a real lifesaver in the event of a serious hard disk crash. With DOS preinstalled on so many computers, taking the time to create an emergency boot disk could pay off later.

Gregory Harris is an editor for The Cobb Group's Corporate Computing *journal.*

WINDOWS WARNING

VERSION 5.0 & 6.0

Adding RAM to your system can slow Windows

By Curt Havlin

If you're thinking about adding more RAM (random-access memory) to your system, you should be aware of a hardware limitation that can cause Windows to run extremely slowly. This limitation affects how the system accesses the additional RAM you install.

If you install more RAM directly onto the motherboard (the circuit board that distributes power to your computer's chips), your system accesses the additional RAM at system speed. For example, if you have a 386 running at 33 MHz, installing memory on the motherboard allows your system to access the additional RAM at 33 MHz.

However, adding RAM to an extended memory card can cause problems. When you install a memory card in an available slot, the system accesses the additional RAM via the memory card at what is known as the *bus speed*. The bus speed is slower than the system speed. On a given system, the bus speed might be as slow as 8 MHz. Using this type of memory card will cause Windows to run very slowly.

On the other hand, some systems use proprietary memory cards that you install in a specifically designed slot. This type of memory card allows the system to access the additional RAM at system speed rather than bus speed. Consequently, Windows won't suffer from any performance degradation when it uses the additional RAM.

So, before you use an extended memory card to add RAM to your system, make sure the system can access the memory at system speed, not bus speed. Doing so will save you both aggravation and money.

Curt Havlin is editor-in-chief of Inside Microsoft Windows, *a Cobb Group journal containing tips and techniques for Microsoft Windows.*

Keeping your system in sync with daylight savings time

This year, daylight savings time begins on Sunday, October 31. If you live in an area that observes this time change, don't forget to set your system clock (as well as your other clocks) back an hour. To reset your system clock, type *time* at the prompt and press [Enter]. DOS responds

```
Current time is currenttime
Enter new time:
```

where *currenttime* is the time your system clock currently reads. At the prompt to enter a new time, type the correct time. You can use the 24-hour format, or you can follow the time you enter with *a* (for AM) or *p* (for PM). For example, to set the time to 10:30 at night, you can type *22:30* or *10:30p*. Note that even though the TIME command displays seconds and nanoseconds, you don't have to specify these when you reset the time.

VERSION 5.0

Creating an advanced RENAME command

By David Reid

In last month's issue of Inside DOS, *we told you how to use DOS 6's MOVE command to rename directories. For those of you who prefer DOS 5, we present a simple DEBUG utility you can use to rename directories.*

Although DOS 5 offers plenty of features that make your work at the DOS prompt easy, it lacks a RENAME command that lets you rename directories and move files. Of course, many utility programs let you do these tasks, but why should you have to spend more money for this seemingly routine capability?

In this article, we'll show you how to create a simple COM file that can rename a directory without affecting the files stored in the directory. In addition, this COM file can rename a file and move it to a new directory, all in one operation.

Introducing RENAME2.COM

You can use the DEBUG script in Figure A to create the program RENAME2.COM. Start by typing *edit* at the DOS prompt to open the MS-DOS Editor. Then, enter the script, save it as RENAME2.SCR, and exit the Editor. (If you prefer, you can use a different text editor as long as it saves the file without any additional formatting.) Once you've created the RENAME2.SCR file, run it through DEBUG by using the command

```
C:\>debug < rename2.scr
```

Doing so creates the RENAME2.COM program.

Once you've created RENAME2.COM, you can begin renaming and moving files. Let's look at some example operations you can perform using this new program. First, let's create a test file and temporary subdirectory by using the following commands:

```
C:\>md testing
C:\>dir > \testing\test.tmp
```

To verify the existence of the TEST.TMP file in the TESTING directory, issue the command

```
C:\>dir testing
```

Now that you've created a test file, let's change its name from TEST.TMP to TESTING.TMP by using our RENAME2.COM program:

```
C:\>rename2 c:\testing\test.tmp c:\testing\testing.tmp
```

To verify the filename change, again issue the command

```
C:\>dir testing
```

Notice that the RENAME2 command takes full pathnames as arguments. Try using this format with the standard RENAME command and you'll get the old familiar error

```
Invalid filename or file not found
```

Figure A

```
n rename2.com
a 100
mov si,0080
mov bx,0081
sub ch,ch
mov cl,[si]
or  cl,cl
jz  0126
add bl,cl
call0135
mov dx,si
call0142
call0135
mov di,si
call0142
mov ah,56
int 21
jnb 0131
mov dx,0056
add dx,0100
mov ah,09
int 21
mov ah,00
int 21
inc si
cmp si,bx
jge 0126
mov cl,[si]
cmp cl,20
jz  0135
ret
inc si
cmp si,bx
jg  0126
mov cl,[si]
cmp cl,0d
jz  0153
cmp cl,20
jnz 0142
mov [si],ch
ret

e 156
72 65 6e 61 6d 65 32
e 15d
20 65 72 72 6f 72 24
r cx
64
w
q
```

This DEBUG script creates RENAME2.COM, which renames files and subdirectories and moves files between subdirectories.

Now let's use RENAME2 to change the name of our temporary subdirectory from TESTING to TESTDIR by entering the command

```
C:\>rename2 \testing \testdir
```

You can verify this directory name change by simply typing *dir testdir* at the prompt.

Finally, let's use RENAME2 to move a file and simultaneously change its name. Use the following command to move the TESTING.TMP file to the root directory and, at the same time, change its name back to TEST.TMP:

```
C:\>rename2 \testdir\testing.tmp \test.tmp
```

Again, verify the move and filename change by typing *dir*.

As you can see, RENAME2 has many advantages over the standard RENAME command. However, RENAME does have one important feature not found in RENAME2: You can pass wildcards to RENAME (*.BAT, *.WPS, and so forth) for renaming multiple files with one command. Our new RENAME2.COM program can rename only one file at a time. By having both RENAME and RENAME2, you have commands to suit all your file and directory renaming needs.

David Reid is editor-in-chief of The DOS Authority, *a Cobb Group publication for advanced DOS users.*

UNDOCUMENTED DOS TIP

VERSION 5.0 & 6.0

Bypassing the *Abort, Retry, Fail?* error message

By Greg Shultz

How many times have you accidentally tried to access a floppy disk drive from the DOS prompt when there was no floppy disk in the drive? When you do, you're confronted with the infamous DOS error message

```
Not ready reading drive X
Abort, Retry, Fail?
```

where *X* is the floppy disk drive letter. At this point, you must type *F* (for Fail) since typing *A* (for Abort) or *R* (for Retry) simply causes DOS to repeat the error message. When you type *F*, DOS responds with the prompt

```
Current drive is no longer valid>
```

and waits for you to type a valid drive letter. Once you do, the DOS prompt returns.

Since there's only one correct reply to this error message and you'll resolve the problem at the prompt that follows the correct reply, typing *F* in the first place could be considered a waste of time. Fortunately, an undocumented COMMAND.COM switch—/F—allows you to bypass the *Abort, Retry, Fail?* error message and go right to the *Current drive is no longer valid* prompt.

As you may know, in order to pass parameters to COMMAND.COM, you must load COMMAND.COM from the CONFIG.SYS file by using the SHELL directive. To configure your system so it automatically bypasses the *Abort, Retry, Fail?* error message, you add the line

```
SHELL=C:\COMMAND.COM  C:\ /F /P
```

to your CONFIG.SYS file and then reboot your system.

When you do, the /F switch automatically responds to the *Abort, Retry, Fail?* error message and brings up the current-drive prompt. Now, if you accidentally attempt to access a floppy disk drive from the DOS prompt when there isn't a floppy disk in the drive, you'll see

```
Not ready reading drive X
Abort, Retry, Fail?Current drive is no longer valid>
```

and you can simply type the correct drive letter.

Notes

In the SHELL directive we just showed you, we used two other parameters that need explanation. In addition to the /F switch, we used the /P switch. You must *always* add the /P switch to the command line when you load COMMAND.COM from the CONFIG.SYS file. This switch tells COMMAND.COM to ignore the EXIT command and remain in memory indefinitely. It also makes COMMAND.COM the first command processor, which means COMMAND.COM loads and processes the AUTOEXEC.BAT file commands. If you don't use the /P switch, COMMAND.COM will ignore your AUTOEXEC.BAT file during bootup. Typing *EXIT* at any time will cause your system to crash, and you'll have to reboot.

You'll also notice that in the SHELL directive, the path to COMMAND.COM is followed by C:\. Known as the comspec parameter, C:\ sets up the COMSPEC environment variable, which DOS uses to specify COMMAND.COM's location on the hard disk. Any executable program (including COMMAND.COM itself) that needs to access COMMAND.COM will look for the COMSPEC environment variable.

You may have already set COMMAND.COM to load from the SHELL directive in your CONFIG.SYS file in order to increase the DOS environment's size. If you have, you'll find a line similar to

```
SHELL=C:\COMMAND.COM /E:xxxx /P
```

in your CONFIG.SYS file, where *xxxx* is a number from 4 to 32,768. (Typically, this number is either 512 or 1,024.) To bypass the *Abort, Retry, Fail?* error message in this case, you simply insert the /F switch in your existing SHELL directive, as in

```
SHELL=C:\COMMAND.COM /E:xxxx /F /P
```

Greg Shultz is editor-in-chief of Inside PC Tools for Windows *and* Inside Norton Desktop for Windows, *two other Cobb Group publications.*

DOS TIPS

VERSION 5.0 & 6.0

Tricking DOS with NUL

By Suzanne Thornberry

You may have noticed that many batch files use NUL, a term DOS reserves to stand for *nothing*. Admittedly, NUL doesn't sound very useful at first, but it can serve a purpose when you want to "fool" DOS. In this article, we'll show you two uses for NUL that can improve batch files and even save you time at the command line.

Checking for the presence of a directory

Have you ever wanted to make a batch file check for a directory before it proceeds? As you may know, you can use the statements IF EXIST and IF NOT EXIST to check for the presence of a file. Although these statements won't work for directory names, you can use NUL as a "dummy" file specification within the directory. As long as the path you specify exists, DOS will proceed as if it found a file named NUL in that directory.

For example, suppose you have a batch file called DITCH.BAT that deletes files to the LIMBO directory on a RAM disk. (For more on RAM disks, see "RAM Disks: Not Quite the Best of Both Worlds," in the December 1992 issue.) When you boot up, your RAM disk is empty. You could create the LIMBO directory by placing in your AUTOEXEC.BAT file the command

```
md d:\limbo
```

(assuming your RAM disk is the D: drive), but it's more appropriate to put the command in DITCH.BAT. However, you don't want DITCH.BAT to issue the MD command every time it runs. After running the batch file for the first time, repeating the MD D:\LIMBO command will generate the error message *Unable to create directory*. The solution is to use the following IF statement to create the LIMBO directory if it doesn't already exist:

```
if not exist d:\limbo\nul md d:\limbo
```

By including this statement, DITCH.BAT will be able to tell if your RAM disk contains a LIMBO directory—*even* if the directory contains no files.

Redirecting screen output to the NUL device

We've shown you how NUL can function as a dummy filename, but NUL more commonly refers to the NUL device. You can redirect output to the NUL device just as you would to any other device, such as PRN, the printer. Redirecting a command's output to NUL effectively hides the output, since the NUL device dumps anything it receives. (For this reason, programmers sometimes refer to the NUL device as a "bit bucket.")

This technique sometimes comes in handy when you're writing batch files. Since a batch file takes control of DOS, users really don't need to see the messages that many DOS commands generate. DOS normally writes these messages to the CON device, which is short for *console*. (CON stands for the keyboard as an input device, or the monitor as an output device.) You can prevent messages from appearing onscreen by redirecting a command's output to the NUL device instead of the default CON.

For example, suppose you use a batch file that copies files from a diskette by using the command

```
copy %1:*.* %2
```

and then displays its own success or failure message. In such a case, you don't need to see DOS display the *File(s) copied* message. To hide this message, use the command

```
copy %1:*.* %2 > nul
```

to copy the files from the diskette instead.

Although the technique of redirecting a command's output to the NUL device is most common in batch files, you can also use the technique from the command line. This is especially useful when you've used [Ctrl][PrintScrn] to echo all screen output to the printer. (Make sure your printer is online before you press this key combination.) You can send selected output to the NUL device in order to save paper. For example, suppose you use the following XCOPY command to copy all files in the current directory (in this case, C:\DATA) to the A: drive:

```
C:\DATA>xcopy *.* a:
```

Normally when you issue the XCOPY command, DOS will display the message *Reading source files* and then list each file that it copies. You can prevent this information from appearing onscreen—and echoing to the printer—by adding > *nul* to the end of the command:

```
C:\DATA>xcopy *.* a: > nul
```

Conclusion

In this article, we've shown you how you can use NUL to trick DOS. Using NUL as a dummy file specification lets you test for the presence of a particular directory. You can also hide the messages generated by a command by redirecting its output to the NUL device.

Suzanne Thornberry is the managing editor of Inside DOS.

DOS BASICS

VERSION 5.0 & 6.0

How to check if the files you're copying will fit on one disk

Don't you hate getting an *Insufficient disk space* error when you're in the middle of copying files to a disk? Of course, the easiest way to avoid this error message is to make sure all the files you're copying will fit on one disk. How? A simple issue of the DIR command will do the trick. When you use the DIR command, DOS not only lists files, but also follows the list with the number of files in the listing and the total number of bytes the files take up.

For instance, let's say you want to store all the batch files in your BATCH directory on a single disk. To find out if they'll fit on one disk, you can switch to the BATCH directory and type *dir*. Your screen will look similar to

```
C:\BATCH>dir

Volume in drive C is DOS-PRIMARY
Volume Serial Number is 1A9D-81D8
Directory of C:\BATCH

.            <DIR>     05-27-93   10:08a
..           <DIR>     05-27-93   10:08a
WHATSIS  BAT      1201 08-02-93    3:00p
KEYPRES  SCR       806 07-30-93    1:43p
TPAUSE   SCR       455 08-04-93    3:47p
TPAUSE   COM       111 08-04-93    3:47p
DOSSHELL BAT       107 01-30-93    3:43p
    .     .         .     .          .
    .     .         .     .          .
    .     .         .     .          .
CHKLIST  BAT       189 04-28-93   11:49a
READ     BAK        82 05-05-93    4:11p
INSTALL  BAK       232 05-19-93    1:40p
     2248 file(s)      568600 bytes
                     16152836 bytes free
```

To make a proper comparison of the number of bytes the files take up and the capacity of your disk, you must know how many bytes your disk can hold. Table A shows this figure for each of the four disk types.

Table A: *Disk capacities*

Disk size	Disk density	Capacity (in bytes)
5.25"	360 Kb	362,496
3.5"	720 Kb	730,112
5.25"	1.2 Mb	1,213,952
3.5"	1.44 Mb	1,457,664

The sample listing takes up 568,600 bytes. Therefore, you need at least a 3.5" 720-Kb disk to fit all the files on just one disk. Now let's suppose the only disk you have on hand is a 5.25" 360-Kb disk. Obviously, all the files in the BATCH directory won't fit on this disk. But do you really need all the files? You'll notice that the sample listing contains files with extensions other than BAT. Since you really want to store only the batch files on the disk (those with BAT extensions), you can check how many bytes just the batch files take up by supplying DIR with the file specification *.BAT:

```
C:\BATCH>dir *.bat

Volume in drive C is DOS-PRIMARY
Volume Serial Number is 1A9D-81D8
Directory of C:\BATCH

.             <DIR>      05-27-93  10:08a
..            <DIR>      05-27-93  10:08a
WHATSIS  BAT        1201 08-02-93   3:00p
DOSSHELL BAT         107 01-30-93   3:43p
WAIT15   BAT          77 08-04-93   3:42p
TODAY_MS BAT         231 01-27-93  11:14a
SOLI     BAT          53 10-03-92   4:12p
    .      .          .      .        .
    .      .          .      .        .
    .      .          .      .        .
ALL      BAT          27 05-04-93   1:51P
READ     BAT          82 05-05-93   4:11p
INSTALL  BAT         232 05-19-93   1:40p
      811 file(s)       223070 bytes
                      16152836 bytes free
```

Since the files with a BAT extension take up just 223,070 bytes, they'll all fit on a 5.25" 360-Kb disk, which can hold up to 362,496 bytes.

A note on disk structure

Although, in theory, a group of files will fit on one disk as long as the total number of bytes doesn't exceed the capacity of the disk, you might occasionally find that this isn't the case. Floppy disk storage units, or *clusters*, are composed of sectors, typically one or two. A sector can hold up to 512 bytes. However, DOS won't allocate less than a full cluster to each file, whether a file fills the cluster or not. Therefore, if each cluster on a disk consists of one sector, a 1-byte file will take up as much room on the disk as a 512-byte file. Likewise, if each cluster consists of two sectors, a 1-byte file will take up as much room as a 1,024-byte file. The bytes a file doesn't use reduce the effective capacity of the disk.

LETTERS

VERSION 5.0 & 6.0

Redirecting output to a file

I have a small batch file that doesn't behave the way I expect it to. I'm trying to redirect output to another file, but for some reason, the file always comes up empty. Here's a sample of what part of my batch file looks like:

```
IF ERRORLEVEL 3 IF NOT ERRORLEVEL 4 ECHO There are 3
    items > TEMP.DAT
IF ERRORLEVEL 2 IF NOT ERRORLEVEL 3 ECHO There are 2
    items > TEMP.DAT
IF ERRORLEVEL 1 IF NOT ERRORLEVEL 2 ECHO There is only 1
    item > TEMP.DAT
```

If I don't redirect the output, I see the appropriate phrase on my screen. For instance, when an ERRORLEVEL of 2 is set, I see *There are 2 items*; if an ERRORLEVEL of 3 is set, I see *There are 3 items*. However, when I redirect the output, I get nothing, unless the ERRORLEVEL is 1. What's going on?

Jerry Johansen
New York, New York

When you use > to redirect output to a file, the output overwrites the current contents of the file. In Mr. Johansen's sample batch file, the output he redirects to the TEMP.DAT file is *nothing* if the conditional statement is false and a phrase if the conditional statement is true.

Therefore, if the first statement is true and the other two are false, the first statement echoes the phrase *There are 3 items* to the TEMP.DAT file; the second statement replaces this phrase with *nothing*—in other words, it empties the TEMP.DAT file.

If the first and third statements are false but the second is true, the first statement sends *nothing* to the file TEMP.DAT, the second statement replaces *nothing* with the phrase *There are 2 items*, and the third statement replaces this phrase with *nothing* again.

Finally, if the first two statements are false but the last is true, the first two statements send *nothing* to the TEMP.DAT file, but the third statement replaces *nothing* with the phrase *There is only 1 item*.

The solution? Append the output to the TEMP.DAT file instead of overwriting the contents. Since only one of the conditions is true at any given time, appending the output will ensure that the appropriate phrase is written to the TEMP.DAT file but not overwritten by the output of false conditions. To append the output, simply replace > with >>. The new batch file will look like

```
IF ERRORLEVEL 3 IF NOT ERRORLEVEL 4 ECHO There are 3
    items >> TEMP.DAT
IF ERRORLEVEL 2 IF NOT ERRORLEVEL 3 ECHO There are 2
    items >> TEMP.DAT
IF ERRORLEVEL 1 IF NOT ERRORLEVEL 2 ECHO There is only 1
    item >> TEMP.DAT
```

Microsoft Technical Support
(206) 454-2030

SECOND CLASS MAIL

Please include account number from label with any correspondence.

If you're familiar with other programming languages, you might be surprised that DOS doesn't simply ignore the rest of a statement when the condition is false. In most conditional statements, it will. However, DOS doesn't ignore redirection symbols, even when they follow a false condition. In fact, redirection symbols are so powerful that DOS won't even ignore them when they occur in a REM statement.

Displaying a batch file screen for a specific time

I write quite a few batch file programs, and one thing I'd like to do is use a timing sequence to display some of the screens in these programs for 15 to 20 seconds. Is there a DOS batch command that will enable me to do this?

Emanuel Greenblatt
Bethpage, New York

If you use DOS 6, the CHOICE command allows you to pause a batch file for a specific amount of time. For example, to display a message for 15 seconds, you'd enter the commands

```
echo message
choice /t:y,15 > nul
```

in the batch file, where *message* is the text you want to display before pausing the batch file. By default, this form of the CHOICE command will cause the batch file to resume if the user presses Y or N. However, the batch file doesn't prompt the user to press a key, so the user probably won't. The /T:Y,15 switch pauses the batch file for 15 seconds, waiting for the user to press a valid key. Since the user probably won't, the batch file will pause for the full 15 seconds before resuming. When the batch file resumes, > NUL—which prevents the valid key choices from displaying onscreen—redirects the default Y response to the NUL device so it doesn't appear onscreen.

If you use DOS 5, you can use the DEBUG script, WAIT.SCR, shown in Figure A, to create the WAIT command. (The original version of WAIT appeared in the April 1993 issue of *The DOS Authority*, a Cobb Group publication for advanced DOS users.) This command simply pauses the batch file for 15 seconds and then resumes execution.

Figure A

```
n wait.com
a 100
mov si,f
mov ax,000f
mul si
mov bx,ax
mov ax,4240
mul si
mov cx,dx
add cx,bx
mov dx,ax
mov ah,86
int 15
int 20

rcx
1b
w
q
```

WAIT.SCR creates WAIT.COM, which includes a built-in timer that continues the batch file's execution after 15 seconds.

After creating the WAIT.SCR file (using the DOS Editor or an ASCII-compatible word processor), you issue the command

```
C:\>debug < wait.scr
```

to create WAIT.COM. Then, wherever you want to pause a batch file for 15 seconds, you just include the WAIT command.

Note: The WAIT command makes use of your computer's BIOS event timer. Since only one program at a time can use the BIOS event timer, make sure you don't try to access it from two batch files that both invoke the WAIT command and run simultaneously in two different DOS sessions initiated from Windows. If you do, the first program to access the BIOS event timer will permanently hang when the second batch file tries to access it.

Printed in the USA
Printed on recycled and recyclable paper
50% recycled fiber including 10% post-consumer waste paper

October 1993
Vol. 4 No. 10

COBB INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS Versions 5 & 6

WHATSIS.BAT displays files to help you decide if you can delete them

VERSION 5.0 & 6.0

Bill Woodruff, a systems editor for Phoenix Newspapers, Inc., developed the batch file we present in this article.

Have you ever stumbled upon a group of similarly named files whose purpose eluded you? Some programs, like Microsoft Works, create such files with names like WK######.TMP (where ###### represents a string of six digits). If you've ever encountered files like this, your first thought might have been, "What's 'is? Temp files. Let's get rid of 'em." If you aren't overly impulsive, your next thought might have been, "Wait. I'd better review them first to see if they contain anything worthwhile." But if you loathe tedious tasks, you may have decided, "Geez, I don't want to enter *type wk######.tmp ¦ more* for every one of these files." And that was the end of that. You simply left the mystery files to clutter up your hard drive.

If you have disk space to spare, you can get away with leaving unnecessary files on your hard drive. But if you'd rather not risk the chance of getting an *Out of disk space* error at a critical moment, your best bet is to periodically rid your hard drive of extra files. In this article, we'll demonstrate a batch file that, with one command, lets you view the contents of all the mystery files and then decide whether to keep them or delete them.

The WHATSIS (What's 'is?) batch file

In last month's *Inside DOS*, Van Wolverton outlined the importance of disk housekeeping and described some of DOS' built-in tools to help you with the work (see "Disk Management: It's High Tech, but It's Still Housekeeping"). When you want to make sure you aren't blindly deleting files you need, however, the WHATSIS batch file is the tool of choice. To create this batch file, shown in Figure A on page 2, use the DOS Editor or another ASCII-compatible word processor to enter the code; then, save it under the name WHATSIS.BAT.

After creating the WHATSIS batch file, you run it by typing

```
C:\>whatsis file specification
```

and pressing [Enter]. The *file specification* should include a path if you aren't checking files in the current directory. It can also include wildcards. For instance, to view the files in the WORKS directory matching the file pattern *WK######.TMP*, you'd enter

```
C:\>whatsis works\wk*.tmp
```

Adjustments to WHATSIS.BAT for DOS 5

You might have noticed that the batch file shown in Figure A uses the CHOICE command found only in DOS 6. If you're a DOS 5 user, you'll need to make a few adjustments before you can use the WHATSIS batch file. Instead of using the CHOICE command, you'll need to use the KEYPRESS.COM program, which allows a batch file

IN THIS ISSUE

A Publication of The Cobb Group

Figure A

```
@echo off
rem Bill Woodruff's WHATSIS.BAT lets a user review the
rem contents of a file or group of files and then delete
rem the files, if desired.
cls
if "%1"=="" goto :WHOOPS
for %%a in (%1) do type %%a >> whatsis.fyl
copy whatsis.fyl whatsis.xxx > nul
del whatsis.fyl
if not exist whatsis.xxx goto :NONE
type whatsis.xxx | more
echo.
pause
del whatsis.xxx
cls
echo WOULD YOU LIKE ME TO DELETE ALL THE FILES
choice MATCHING THE SPECIFICATION %1 /C:yn
if errorlevel 2 goto :SKIP
if errorlevel 1 goto :KILL

:WHOOPS
echo PLEASE RUN WHATSIS.BAT AGAIN, USING THE SYNTAX
echo.
echo whatsis path\filename.ext
echo.
echo.
echo THE FILENAME AND EXTENSION MAY INCLUDE WILDCARDS
echo SUCH AS
echo.
echo whatsis works\*.tmp
echo.
echo OR
echo.
echo whatsis works\wk*.tmp
echo.
echo OR
echo.
echo whatsis works\wk*.*
echo.
echo.
echo BATCH FILE TERMINATING
goto :END

:NONE
echo NO FILES MATCHING THE SPECIFICATION %1
echo WERE FOUND
goto :END

:SKIP
echo.
echo.
echo NO FILES DELETED
goto :END

:KILL
dir %1 | find "(s)" > zzyyxxq.xxx
echo          DELETED >> zzyyxxq.xxx
del %1
echo.
echo.
type zzyyxxq.xxx
del zzyyxxq.xxx > nul

:END
```

The WHATSIS batch file ensures you aren't deleting files you really want to keep.

INSIDE DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices

	Domestic:	$49/yr. ($6.00 each)
	Outside US:	$69/yr. ($8.50 each)

Phone

	Toll free:	(800) 223-8720
	Local:	(502) 491-1900
	Customer Relations Fax:	(502) 491-8050
	Editorial Department Fax:	(502) 491-4200

Address

You may address tips, special requests, and other correspondence to

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster

Second class postage is paid in Louisville, KY. Send address changes to

Inside DOS
P.O. Box 35160
Louisville, KY 40232

Back Issues

To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $6.00 each, $8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all the articles that have appeared in *Inside DOS*.